PCT/EP03/03551

10/510353

#2

PA 998080

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

April 22, 2003

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***60/370,608***
FILING DATE: ***April 05, 2002***

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

CERTIFIED COPY OF
PRIORITY DOCUMENT

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

P. R. GRANT
Certifying Officer

04-09-02

A/PROV

JC845 U.S. PTO
04/05/02

Please type a plus sign (+) inside this box → [+]

PTO/SB/16 (8-00)
Approved for use through10/31/2002 OMB 0651-0032
U S. Patent and Trademark Office, U S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

JC973 U.S.PTO
60/370608
04/05/02

INVENTOR(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Klaus	Hoffesommer	Lichtenfels, Germany

☐ *Additional inventors are being named on the ____ separately numbered sheets attached hereto*

TITLE OF THE INVENTION (280 characters max)

OPTIMIZATION OF COLLATERALIZED LOANS

Direct all correspondence to: **CORRESPONDENCE ADDRESS**

☐ Customer Number [ ] → *Place Customer Number Bar Code Label here*

OR *Type Customer Number here*

☒ Firm or Individual Name: CASELLA & HESPOS LLP

Address: 274 Madison Avenue

Address: Suite 1703

City: New York | State: New York | ZIP: 10016

Country: USA | Telephone: (212) 725-2450 | Fax: (212) 725-2452

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Specification *Number of Pages* 19 ☐ CD(s), Number

☒ Drawing(s) *Number of Sheets* 12 ☐ Other (specify)

☒ Application Data Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT *(check one)*

☒ Applicant claims small entity status. See 37 CFR 1.27.

☒ A check or money order is enclosed to cover the filing fees

☒ The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number 03-1030

☐ Payment by credit card. Form PTO-2038 is attached.

FILING FEE AMOUNT ($): $80.00

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U S. Government agency and the Government contract number are: _____

Respectfully submitted,

SIGNATURE Gerald E. Hespos

Date 4-5-02

TYPED or PRINTED NAME Gerald E. Hespos

REGISTRATION NO. *(if appropriate)* 30,066

Docket Number: H 2201

TELEPHONE (212) 725-2450

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is used by the public to file (and by the PTO to process) a provisional application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the complete provisional application to the PTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, D C.

P19SMALL/REV05

CERTIFICATE OF MAILING BY "EXPRESS MAIL" (37 CFR 1.10) Applicant(s): Klaus Hoffesommer			Docket No. H 2201
Serial No. --	Filing Date Concurrently herewith	Examiner -	Group Art Unit --

Invention: OPTIMIZATION OF COLLATERALIZED LOANS

I hereby certify that the following correspondence:

U.S. Provisional Patent Application

(Identify type of correspondence)

is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 in an envelope addressed to: The Assistant Commissioner for Patents, Washington, D.C. 20231 on

April 5, 2002
(Date)

Marie B. Bufalo
(Typed or Printed Name of Person Mailing Correspondence)

Marie B. Bufalo
(Signature of Person Mailing Correspondence)

EV092895943US
("Express Mail" Mailing Label Number)

Note: Each paper must have its own certificate of mailing.

Inventor Information

Inventor One Given Name::	Klaus
Family Name::	Hoffesommer
Postal Address Line One::	Am Klapperkump 5a
City::	Lichtenfels
Postal or Zip Code::	35104
Citizenship Country::	GERMANY

Correspondence Information

Correspondence Customer Number::	001218
Telephone::	(212) 725-2450
Fax::	(212) 725-2452
Electronic Mail::	email@casella-hespos.com

Application Information

Title Line One::	Optimization of Collateralized Loans
Total Drawing Sheets::	12
Formal Drawings?::	Yes
Application Type::	Provisional
Docket Number::	H 2201

Representative Information

Representative Customer Number::	001218

INVENTOR : KLAUS HOFFESOMMER

ATTY. REF.: H 2201

OPTIMIZATION OF COLLATERALIZED LOANS

1. Background of the Invention

If each collateral security has to be assigned to distinct loan accounts and if the loan accounts have to be balanced in an order that should be determined, then the amount of unsecured total credit may vary significantly depending on that order.

Example: let L_1, L_2, ... L_5 be five loan accounts. And let S_1, S_2, ... S_4 be collateral securities allocated to the loan accounts as indicated here in Fig.1 :

Fig. 1 shows that each of these four securities are assigned to distinct loan accounts. If we would offset the securities against the loan accounts in the following order

$L_1 - S_1$, $L_2 - S_1$, $L_2 - S_2$, $L_3 - S_3$, $L_3 - S_4$ (no further loan account can be balanced)

then on the one hand a total amount of 1 million $ would remain unsecured (total share of "blank" credit) - and on the other hand 0.9 million $ of security value could not be offset against the assigned loan accounts. But if the securities are offset in an optimized order, for example:

$L_5 - S_1$, $L_4 - S_4$, $L_3 - S_3$, $L_3 - S_1$, $L_2 - S_2$, $L_2 - S_1$, $L_1 - S_1$

the total amount of unsecured credit would be minimized (0.1 million $).

Fig. 2 represents the above assignments as a network. Small networks like this one may easily be optimized without a computer program. But when managing collateralized loans of commercial enterprises the related networks often consist of several hundred (and sometimes several thousand) assignments. Here the order in which these securities are to be offset against the assigned loan accounts can only be optimized by a computer program. Fig. 3 gives an idea of a network related to collaterlized loans of a commercial enterprise.

2. Summary of the Invention

Optimization of the order of securities assignments is provided by a system that consists of two parts, a network flow method that results in an optimal flow and a method that derives the optimal order from this flow.

2.1 the network flow method

The flow optimization method results in an optimal flow of capital (credit) from vertices (or nodes) representing the loan accounts to the vertices representing collateral securities. A short overview is given here to explain this method.

After a network builder has built up a network that can be represented by a directed bipartite graph including a source vertex s and a sink vertex t (see Fig. 4), and after the optimal value of the total flow that should reach the sink vertex has been ascertained by an s-t-cut of minimum capacity, a special network preflow-push method sends a maximum amount of flow from a source vertex s to a sink vertex t. It works by pushing flows along individual edges of the network, each flow respecting the capacity constraints at the respective edge, resulting in successive preflows until an optimal flow can be obtained. This network flow method is suited for bipartite graphs, and it is suited especially to the task of determining an optimal order of the assignments, hence superseding other methods for optimization in permutation spaces ([1]) by a preflow-push optimization method with an incomparably better algorithmic efficiency. Thus the goal of this method is not the optimal flow but an optimal order (i.e. to derive an optimal permutation from the network flow, a permutation that would lead just to this distribution of flows). The preflow-push method is suited to this goal by respecting additional constraints defined for all pushes of flows along the edges, by handling reflows at each edge the same way as the preceding flows, and by applying a special label method. So the characteristic feature of this optimization system is a network flow design that enables to derive the corresponding optimal permutation.

general flow characteristics:

During the preflow stage of the flow method - while each preflow respects the capacity constraints at each edge - excess flow may occur at the vertices of the network: the flows entering a vertex (inflows) may exceed the flows leaving the respective vertex (the outflows).

The resulting network preflow is optimal if the amount of flow that reaches the sink corresponds to the value ascertained before by the min s-t-cut. Then the preflow-push algorithm stops - the flow is optimal because there is no further augmenting path (from the source to the sink). The method then converts the preflow into a flow by reducing the excess flow at each vertex, so that inflows = outflows at every vertex (except for source and sink).

---

1 The amount of credit balanced by a collateral security assigned to a loan account depends on the order in which the securities are offset against the assigned loan accounts. The total amount of unsecured credit depends on this order as well - as I have shown in chapter 1: it is only the order of these assignments that determines the total amount of unsecured credit.
As Silvio Turrini (*1, 1996, digital, Western Research Laboratory) has pointed out, optimization problems usually deal with sets of independent values. In this case however the order of assignments "constitutes the n-tupla of values" and it is this order that "differentiate one input from another and the value of any parameter at a given position in the n-tupla is clearly dependent on all the others" (*1, p.1). The method presented here is an extremely fast algorithm that may generally replace the slow algorithms dealing with permutation spaces in different other ways.

The final optimal flow meets the following conditions:

1. The resulting network flow respects the capacity constraints at each edge. (This condition has already been respected during the preceding preflows)
2. for all vertices of the network (except source and sink) inflows = outflows
3. There is no further augmenting path from the source to the sink.

The network represented by the graph of **Fig. 4** would have been constructed by the network builder for the collateralized loans of Fig.1. The capacities of the edges are indicated in million US $. Please note that the method described here operates with integer values - in this case with cents rather than with US $.

Fig. 5 is a representation of a bipartite graph for a typical network of collateralized loans of a commercial enterprise. These graphs tend to be strongly connected.

2.2 deriving the optimal order from the values of the network flow

The last stage of the optimization method is an algorithm that derives the optimal order from the capital flows on each edge. The system obtains the optimal order by offsetting the securities against the loan accounts in an order that corresponds exactly to the values of the network flow. This is done by an iteration that chooses the edges representing the credit-security assignments in such an order that at each step of the iteration the values of the edge flows correspond to the values that will be obtained when offsetting the securities against the loan accounts exactly in this order.

3. Technical Description

3.1. The Network construction algorithm

The optimization system first constructs a directed graph. This "network construction algorithm"

1. begins selecting a single loan account, then - in an iterative process
2. it selects all securities assigned to all those loan accounts that are not yet searched for assigned securities - and establishes the edges that link these securities to the loan accounts
3. selects all loan accounts assigned to all those securities that are not yet searched for assigned loan accounts - and establishes the edges that link these loan accounts to the assigned securities
4. continues with step 2 - 3 until no more loan accounts and no more securities can be found that are linked to that network

At this stage the graph corresponds to the graphs shown in Fig.2 and and Fig.3 (p.4).

The construction algorithm then completes the directed graph. It assigns the capacities to each edge:

1. It implements a source vertex (s), creates the edges from the source vertex to all loan account vertices and assigns capacities to each edge: The value of the capacity of each edge leaving the source vertex and entering a loan account vertex is the corresponding amount of credit.

2. then the construction algorithm assigns capacities to all edges that leave the loan account vertices and enter the security vertices. The capacity of each of these edges is the minimum value of either of the two corresponding vertices - either the amount of credit or the value of the assigned security.

3. Finally the algorithm implements a sink-vertex (t), creating edges from all security vertices to the sink-vertex and it assigns capacities to each of these edges. The capacity of each edge leaving the security vertex and entering the sink-vertex is the value of the corresponding security - represented as an integer value.

Fig.5 shows a graph constructed at this stage. The representation of the small graph of Fig.4 also displays the capacities.

The flow method operates on integer values, so all capacities and every flow is represented in this format.

3.2. implementing the bipartite graph: technical features

The bipartite graph is implemented by means of two- and three-dimensional tables ([2]), linked with pointers. Some details are explained here.

3.2.1 The loan account vertices

A table is implemented for the loan account vertices. Each row represents one loan account vertex and has the following attributes / values:

proper values of the vertex
- the identification code of the loan account vertex,
- the amount of credit (integer value),

values referring to the edges
- number of edges leading to coll. security vertices,
- imbedded table with pointer to the edges leading to coll. security vertices,

total flow values
- S-L-flow: flow on the edge from the source vertex to the current loan account vertex
- L-S-flow: total flow from current loan account vertex over all edges leading to security vertices (reflows are always subtracted from flow)
- *remark*: the excess flow at a loan account vertex is inflow - all outflows of the current loan account vertex. That is of course S-L-flow minus L-S-flow

"distance" label
- a label (small integer) is initially set to the value 2, as the distance of the loan account vertex to the sink. During the course of the preflow method, all "distance"-labels are updated and they control the flows in such a way that the flows correspond to the effects of a coherent quick and effective determination of corresponding permutations (and their calculation)

value for the derivation of the optimal order (last stage of optimization method)
- unsecured amount of credit. Initially this is the total amount of this loan account

3.2.2 The collateral security vertices

Each row of this table comprises the values of one collateral security vertex:

proper values of the vertex
- the identification code of the collateral security vertex,
- the value of the security (integer value)

values referring to the edges
- pointer to the first edge that enters the current vertex (edges leaving security vertices)

values referring to the flow
- L-S-flow: total flow on all edges from loan account vertices to the current security vertex
- s-t-flow: flow on the edge from current security vertex to the sink-vertex
- pointer to the corresponding row of the table of active security vertices

2 These tables are of fixed maximum size in order to avoid computer memory allocation at any point during the course of the method

- *remark* : the excess flow at a collateral security vertex is inflow minus all outflows of the current security vertex. That is of course L-S-flow minus s-t-flow

"distance" label

- a label (small integer) that initially is set to the value 1 which is the distance of the coll. security vertex to the sink. During the course of the preflow method, all "distance"-labels are updated and they control the flows in such a way that the flows correspond to the effects of a coherent quick and effective determination of corresponding permutations (and their calculation)

value for the derivation of the optimal order (last stage of optimization method)

- value of the security not yet balanced with loan accounts. Initially this is the total amount of the value of this collateral security

3.2.3 edges loan account vertices → security vertices

Each row of this table represents one edge that leaves a distinct loan account vertex and enters a distinct security vertex and comprises its values:

proper value of the edge

- capacity of the current edge

values referring to the vertices

- pointer to the loan account vertex from which the current edge is leaving
- pointer to the security vertex that is entered by this edge

flow values

- flow on this edge from the loan account vertex to the security vertex (reflows are subtracted. All flows are non-negative)

value referring to the optimal order

- pointer to the corresponding row of the table in which the loan account → security vertex edges are entered in correspondence with the optimal order derived from the network flow. This attribute is managed by that part of the method that derives the optimal order from the flow. The pointer is set when the flow on this edge has been balanced with the loan account vertex. The pointer is null otherwise.

3.3. tables to manage network flow and ascertainment of the optimal order

3.3.1 two tables: activation of loan account vertices / coll.security vertices

- pointer to the corresponding loan account / security vertex
- logic value set to 'true' if vertex is set active
- excess flow of loan account / security vertex

the total number of active loan account vertices and the number of active security vertices is always updated if a vertex is set active / non active

3.3.2 main result table: optimal order of loan account → security assignments

- pointer to the edges linking the loan account vertices to the security vertices
  The pointers are entered (from the first to the last row in this table) in the optimal order of the assignments they are pointing to.

3.4 s-t-cut of minimum capacity

An abridged method of 's-t-cut of minimum capacity' is applied to find a maximum value that the network flow cannot exceed ([3]). This value is the minimum of the total capacities of each of the following cuts, where the "total capacity" of a cut is the sum of the capacities of the edges "leaving" these cuts in this directed graph (directed from the source to the sink vertex):

3.4.1. the minimum of all 'vertical' cuts (see Fig. 6), i.e.
- 3.4.1.1. the total value of loan accounts,
- 3.4.1.2. the total value of all collateral securities
- 3.4.1.3. the total value of all loan account → security assignments

3.4.2. the minimum of all cuts shown in Fig. 7, each cut excluding one more security vertex with an edge leaving the cut from the security vertex toward the sink - from top to bottom. Then each of these cuts is continued as indicated in Fig. 9 (for cut 1), each continuation including one more loan account vertex with an edge leaving the cut from the source to the included loan account vertex from top to bottom

3.4.3. the minimum of all cuts shown in Fig. 8, each cut excluding one more loan account mode with an edge leaving the cut from the source to the loan account vertex - from top to bottom. Then each of these cuts is continued, each continuation including one more security vertex with an edge leaving the cut from the security vertex to the sink - in inverse proportion to the pattern of Fig. 9 (from top right-hand to bottom left-hand)

3.4.4. the minimum of all cuts as explained in 3.4.2 and 3.4.3 - but with progressive exclusion / inclusion from bottom to top.

I will call the minimum value (of capacity) of all these cuts the "*min s-t-cut value*".

---

3 An s-t-cut is a partition of the set of vertices into two subsets such that the source vertex s is member of one subset and the sink vertex t is member of the other subset. To determine the capacity of an s-t-cut we focus on the subset that include the source and there we only consider the capacity of the edges that leave this subset (directed toward the subset that includes the sink). The capacity of a cut is the sum of the capacities of these edges. And the s-t-cut of minimum capacity would always ascertain the value of an optimal flow. But as it is not the flow that shall be determined (but an optimal order instead), the method doesn't process all s-t-cuts and it ascertains a value that is greater or (in most cases) equal to a maximum flow - and this value is one criterion that ends the preflow stage of the flow method.

3.5 the network preflow

3.5.1 standard rules and terminology

The following rules are standard rules for the network preflow-push algorithm (s. Cherkassky /Goldberg *3, Ahuja/Magnanti/Orlin *5 and Korte/Vygen *6)

3.5.1.1 On each edge the flow has to be less or equal to the capacity of the edge:

Let E be the set of edges of graph G, let e denote any edge of this set, let V be the set of vertices of graph G, let v denote any vertex of graph G, let $f(e)$ denote the current flow on edge e, and let $u(e)$ be the capacity of edge e, then

- $f(e) \leq u(e)$ for all $e \in E(G)$

3.5.1.2 During the preflow stage of the method the flows "entering a vertex" may exceed the flows leaving the vertex (in direction source → sink). So let v be a vertex in V, let $\delta^-(v)$ be the number of edges entering a vertex v (in-degree), let $\delta^+(v)$ be the number of edges leaving the vertex v (out-degree), and let s denote the source-vertex, let t denote the sink, then the excess flow =

$$\sum_{e \in \delta^-(v)} f(e) \; - \; \sum_{e \in \delta^+(v)} f(e) \; \geq \; 0 \qquad \text{for all } v \in V(G) \setminus \{s\}$$

(see *6)

3.5.1.3 active vertices

active vertices are those vertices with excess flow (see 3.5.1.2). The discharge operation manages the flows from active vertices. Let v be an active vertex and let w be an adjacent vertex (with at least one edge e' leaving v and entering w): If an edge e' leaving the vertex v is "admissible" (depending on the distance labels of v and the adjacent vertex w - see 3.5.1.4), the discharge operation pushes (excess) flow according to the capacity of this edge.

3.5.1.4 discharge operation: admissible edges

The discharge operation pushes flows from active vertices along admissible edges in the residual graph. The term "admissible" is defined here: If ψ is the labeling function for V(G), then an edge $e(v, w)$ [4] is admissible if $\psi(v) = \psi(w) + 1$ (s. *6, p. 164).

4 leaving vertex v and entering vertex w

3.5.2 special rules applied to the preflow

3.5.2.1 special flow method: general remarks

The standard preflow-push method is designed to push flows in a complex network. In such networks the flow is controlled by distance labels in order to find the shortest viable path from the source to the sink: the flows are pushed "to the nodes that are closer to the sink. If the active node we are currently considering has no admissible arc, we increase its distance label (...)" (*5, p. 224, [5])

But in the preflow-push method presented here flows are pushed along the edges of a bipartite graph in order to find an optimal permutation. Flows may be pushed "forwards" as well as "backwards" at (admissible) edges, and later forwards again on the same edge - updating "distance" labels if no admissible edge exists. At the start of the method these labels really are "distance" labels, but in the course of the discharge operations the labels turn to be constraints that guarantee that different permutations will occur. This goal to produce a flow with the effect of calculating permutations is also supported by the definition of priority-rules for each flow pushed, rules that give priority to one or several edges out of all admissible edges leaving a certain vertex (s. 3.5.5.3.1.1, 3.5.5.3.2.1).

3.5.2.2 termination of the flow method

The preflow stage of this method terminates when the flow to the sink has reached the value of the min s-t-cut - hence the optimal value. In most cases there are still active vertices when the flow method ends - and that is an important difference to standard preflow methods. In a standard network preflow-push method "the algorithm terminates when the network contains no active node" (*5, p.225, [6]). In those standard algorithms "the presence of active nodes indicates that the solution is infeasible" *5, p.224. But as the method for bipartite graphs presented here produces flows that correspond - at different stages of the preflow method - to the effects of a permutation, excess flows are sent back to the source only during the operation that follows the network flow - during the process of deriving the optimal order from the flow.

3.5.3 flow operations

Two important operations are applied iteratively until the optimum is achieved. The discharge operation is the main operation performed for all active vertices, it performs push and relabel operations. The discharge operation is performed iteratively for both sides of the graph:

- 1st for all active loan account vertices ("left" side of bipartite graph) - and
- 2nd for all security vertices ("right" side of bipartite graph)

The discharge operation is first performed for loan account vertices and tries to eliminate all excess flows here by pushing the flows to the coll. security vertices. These pushes eventually lead to excess flow at the security vertices.
Now the flow method looks up for the security vertices that have excess flow and register these vertices as active vertices. In the next phase the discharge operation operates on the active security vertices and pushes excess flow back to the loan account vertices (reflow). After a reflow a part of the loan account vertices will have excess flow again : They are activated and again 'discharged'.

5 please note the equivalence of terms: "node" = "vertex", "arc" = "edge"
6 The reason for this different criterion for termination is the fact that with those standard methods excess flow should flow back to the source in the normal course of the flow method. But the method presented here controls flows and distance labels accordingly to the effects of a permutation. Excess flow should only be sent back while deriving the optimal order.

3.5.4 relabeling

If during a discharge operation the excess flow of a vertex cannot be pushed because there are no (more) admissible edges (s. 3.5.1.4) that leave an active loan account vertex or that enter an active security vertex, the active vertex is relabeled: The label of this vertex is then updated to: 1 + the smallest label of all vertices (except source and sink) that are "linked" to this vertex by an edge with non zero residual capacity (see Fig. 10).

If the label value of some vertex is greater than the number of all vertices of the network, then a decision has to be made: Let N(G) be the total number of vertices of the network and let N'(G) be the number of vertices with those high label values. If $N'G > 1/5 * N(G)$ then the labels of all vertices ('linked' by edges with a non zero capacity) are reset to their initial value: 2 for loan account vertices, 1 for security vertices ("global relabeling"). Otherwise if $N'G \leq 1/5 * N(G)$, then the vertex with such high label value is set 'not active'.

3.5.5 the flow method: technical description

This description of the flow method is based on the preceding definitions and descriptions of important operations ([7]).

3.5.5.1 preparing the network flow

Before the flow method starts working the system prepares the network:

- initial labeling: the labels are set to the distance of the vertices from the sink. As this flow method will never push a flow back to the source (see 3.5.2.2 and note *5) the source vertex is not labeled. The labels of all loan account vertices are set to 2, the labels of the security vertices are set to 1.
- Initial flows are pushed from the source vertex to the loan account vertices according to the capacity of the edges. All loan account vertices (that now have excess flow) are set active and registered in the activation table for loan account vertices (pointers are set on both sides). The values of the excess flows at the the loan account vertices are - at this stage - the flow that has been pushed from the source vertex. These excess flows are entered in the corresponding attribute of the activation table.
- Now the min s-t-cut is ascertained (see 3.4)

3.5.5.2 The flow method: termination criterion

The flow method operates until the flow that has reached the sink corresponds to the value ascertained by the min s-t-cut (see 3.5.2.2). But as an abridged method of 's-t-cut of minimum capacity' has been applied there are rare cases where the s-t-cut of minimum capacity has not been found so that the s-t-cut value might be greater than the value of the optimal flow. In these rare cases other conditions would terminate the flow: The flow method will also end if either there is no more active vertex ([8]), or if the number of total iterations has reached a value that is defined by:

[number of loan account → security edges with non zero capacity] : 4 ([9]), = max. total iterations, with a lower limit of 10. So

3.5.5.2.1 the network flow method operates iteratively until
1st condition: the flow to the sink vertex = min s-t-cut value or
2nd condition: there is no more active vertex or
3rd condition: the number of iterations has reached the maximum value (see above def.).

7 especially 3.5.1.2, 3.5.1.4, and 3.5.3
8 That is exactly the criterion for standard preflow-push methods. But with this method that would only happen (during the course of the flow method) if no credit remains unsecured.
9 experience has shown that in the most complicated cases the optimum has been reached with no more than 1/5 of this value

3.5.5.3 preflow-push method: iterative operation

The network flow operations are repeated iteratively, first for all active loan account vertices (vertices with excess flow), pushing excess flow along the edges to security vertices, then for all active security vertices (to operate the reflow of the excess flow at security vertices), then the process is repeated for the now active loan account vertices (... etc.). The flow method operates the following actions as iterative operation until the conditions for termination explained in 3.5.5.2 are met :

1st repeated discharge operation for all active loan account vertices: see 3.5.5.3.1
2nd registration in activation table for security vertices:
all security vertices "receiving" a first time non zero flow (a flow pushed along an edge from some loan account vertex) are registered in the activation table for security vertices (pointers are set on both sides). The vertices are set 'not active' by default.
3rd push from security vertices to the sink:
excess flow from security vertices is pushed to the sink along all edges with a non zero residual capacity
4th activation of security vertices (for reflow):
all collateral security vertices with excess flow (because inflows to some security vertex were greater than the capacity of the edge of this vertex to the sink) are set active. The excess flow of each vertex is updated in the corresponding row of the activation table.
5th repeated discharge operation for all active security vertices (reflow) : see 3.5.5.3.2
6th activation of loan account vertices:
all loan account vertices with excess flow (after the reflow from coll. security vertices) are set active and the excess flow is updated in the corresponding row of the activation table

3.5.5.3.1 discharge operation for all active loan account vertices

The discharge operation is repeated iteratively for all active loan account vertices until there are no more active loan vertices. The discharge operation would also end if the operation has been performed for all active loan account vertices without any push of excess flow to the security vertices (only relabeling has been done) - under the additional condition that all vertices have already been relabeled, i.e. when the smallest value of all labels of active loan accounts is greater than the highest label of all those security vertices that are linked to the active loan vertices by edges with a non zero residual capacity.

iteration: discharge for active loan account vertices, repeated until no more vertex is active or no push can be performed after each active vertex has been relabeled

operation repeated for every (still) active loan account vertex:

for every active loan account vertex the method performs the following actions at each edge leaving the current vertex and leading to a collateral security vertex - in order to push excess flow :

- it selects a next edge with non zero residual capacity leading to a security vertex, respecting a special order of selection: see 3.5.5.3.1.1 push: special constraint on selection of edges
- if the edge is admissible (s. 3.5.1.4), then excess flow is pushed according to the free capacity of this edge: see 3.5.5.3.1.2 push operation.

if there are no (more) admissible edges and if there is still excess flow the vertex is relabeled (s. 3.5.4 relabeling)

3.5.5.3.1.1 **special constraint on selection of edges for push from loan account vertex**

Special rules are defined for the selection of admissible edges from loan account vertices to security vertices:

If the excess flow to be pushed from an active loan account vertex is greater than the residual capacity of some current edge e
- where residual capacity of an edge e defined as: $u(e) - f(e)$, edge $e \in E(G)$,
($f(e)$ = flow that has already been pushed along this edge) -

and if there is another admissible edge leaving this vertex with a residual capacity ≥ excess flow, then the flow will be pushed along the edge with sufficient residual capacity. i.e. :

Let $ex_f(v)$ be the excess flow of vertex v,

If $ex_f(v) > (u(e) - f(e))$ for an edge $e \in E(G)$
then an edge $e' \in E(G)$ leaving the vertex v has priority, if
$ex_f(v) \leq (u(e') - f(e'))$

3.5.5.3.1.2 **push operation: loan account vertices → security vertices**

Let v be the the active loan account vertex, let e be the current admissible edge that leaves v and enters a security vertex w, let $ex_f(v)$ be the excess flow of vertex v, let $u(e)$ be the capacity on edge e, let $f(e)$ ($f(e) \geq 0$) be the flow that has already been pushed along edge e, let the residual capacity on edge e be $u(e) - f(e)$. Let a := a + b denote an increase of the value of a by the value of b; then the new total flow on edge e would be

$$f(e) := f(e) + ex_f(v) - max (0 ; ex_f(v) - (u(e) - f(e)))$$

i.e. the flow on edge e is to be increased by the excess flow of v if $ex_f(v) \leq (u(e) - f(e))$, otherwise $f(e)$ is to be set to $u(e)$.

In the push operation the method has to increase the flow on edge e and to decrease the excess flow of vertex v :

If $ex_f(v) > (u(e) - f(e))$ then 1. $ex_f(v) := ex_f(v) - u(e) + f(e)$
2. $f(e) := u(e)$

else 1. $f(e) := f(e) + ex_f(v)$
2. $ex_f(v) := 0$
3. vertex v is set not active

3.5.5.3.2 **discharge operation for all active security vertices (reflow)**

A flow on an edge of a directed graph is always considered in the direction that has been defined for the graph. Each flow must be non negative: $f(e) \geq 0$ for all $e \in E(G)$. For this reason the amount of reflow at each edge can never be greater than the amount of the flow that had been pushed at this edge. In this method a reflow is always a flow from a security vertex back to a loan account vertex. Let $f^{-}(e)$ be the reflow on edge e. Then

$f^{-}(e) < 0$ and $f^{-}(e) \geq f(e) * -1$

The discharge operation for active security vertices has to "push back" the excess flow at these vertices according to the "existing" flows on the edges entering these vertices (leaving the loan account vertices):

The discharge operation is repeated iteratively for all active security vertices until there are no more active security vertices. The discharge operation would also end if the operation has been performed for all active security vertices without any push of excess flow (as reflow) to the loan account vertices (only relabeling has been done) - under the additional condition that all vertices have already been relabeled, i.e. when the smallest value of all labels of active security vertices is greater than the highest label of all those loan account vertices that are linked to the security vertices by edges with a non zero flow.

iteration: discharge for active security vertices, repeated until no more vertex is active or no push can be performed after each active vertex has been relabeled

operation repeated for every (still) active security vertex:

for every active security vertex the method performs the following actions at each edge that enters the current vertex (leaving a loan account vertex) - in order to push "back" excess flow :

- it selects a next edge with non zero flow for a push back to a loan account vertex, respecting a special order of selection: see 3.5.5.3.2.1 push: special constraint on selection of edges
- if the edge is admissible (s. 3.5.1.4), then excess flow is pushed "back" according to the flow of this edge: see 3.5.5.3.2.2 push operation.

if there are no (more) admissible edges and if there is still excess flow, the vertex is relabeled (s. 3.5.4 relabeling)

3.5.5.3.2.1 **special constraint on selection of edges for reflow from a security vertex**

Special rules are defined for the selection of admissible edges for the reflow from security vertices to loan account vertices:

1st a flow on an edge leaving a loan account vertex that has no other edge leaving this loan account vertex and leading to another security vertex (i.e. the loan account is assigned to only one collateral security) is never pushed back.

2nd If the the flow $f(e)$ of the current edge e (that is the flow that had been pushed along this edge from the loan account vertex) is equal to the capacity of the edge, i.e.

for a security vertex v and an admissible edge $e \in \delta^-(v)$,

if $f(e) = u(e)$

then - under the following conditions - other edges $e' \in \delta-(v)$ would have priority:

(*1) If there is another admissible edge e' from some loan account vertex to the security vertex v where $f(e') < u(e')$
and $f(e') \geq ex_f(v)$
then the excess flow of vertex v would be pushed ("back") along this edge.

(*2) If there are other admissible edges to this security vertex where for each edge e'

$f(e') < u(e')$

and where

$$\sum_{e' \in \delta-(v)} f(e') \geq ex_f(v)$$

then the excess flow would be pushed ("back") along these edges (from the current active security vertex v)

3.5.5.3.2.2 **push operation: security vertices → loan account vertices (reflow)**

Let v be the the active security vertex, let e be the current edge that leaves a loan account vertex v' and enters v, let $ex_f(v)$ be the excess flow of the security vertex v, and let $f(e)$ ($f(e) > 0$) be the flow that had been pushed from v' along edge e.

The push operation will then decrease the flow on edge e (or "increase by a negative flow") and it will decrease the excess flow of vertex v accordingly :

If $ex_f(v) > f(e)$ then
1. $ex_f(v) := ex_f(v) - f(e)$
2. $f(e) := 0$

else
1. $f(e) := f(e) - ex_f(v)$
2. $ex_f(v) := 0$
3. vertex v is set not active

3.6 deriving the optimal order from the values of the network flow

(technical description)

The system derives the optimal order of assignments by offsetting the securities against the loan accounts corresponding to the values of the network flow.

Let e be a current edge from a loan account vertex to a security vertex and let $f(e)$ be the flow on this edge. Let e' be the edge from the source vertex to the current loan account vertex and let e'' be the edge from the current security vertex to the sink. Let $f(e')$ be the network flow of edge e', let $f'(e') = f(e') - f(e)$ = the reduced flow on e' after a security has been offset against a loan account vertex at edge e, let $f(e'')$ be the network flow of edge e''. Let v be the current security vertex, let $U(e'')$ be the security value (capacity of edge $v \rightarrow$ sink) and let $U'(e'')$ be the "rest" of the security value after it has been offset against a loan account.

First $f'(e') := f(e')$ for all edges from the source to the loan account vertices
and $U'(e'') := U(e'')$ for all security vertices

The order of assignments will be determined:

- All edges (assignments) with flows that meet the following conditions:
  $f(e) = u(e)$ (i.e. flow = capacity of the edge) and
  $f(e) = f(e')$, (i.e. the amount of credit of a single loan account is offset against only one single security)

  For each edge that meet these conditions the assignment is set to the next position and the amount of credit of the loan account is offset against the security, i.e.
  $f'(e') := f'(e') - f(e)$
  $U'(e'') := U'(e'') - f(e)$

- All edges (assignments) with flows that meet the following conditions:
  $f(e) = u(e)$ (i.e. flow = capacity of the edge) and
  ($f(e) = U'(e'') = f(e'')$ or
  $f(e) > U'(e'')$ and $ex_f(v) \geq f(e) - U'(e'')$)

  For each edge that meet these conditions the assignment is set to the next position. If there has been excess flow, the excess flow is sent back to the source. Then the amount of credit of the loan account is offset against the security, i.e.
  $f'(e') := f'(e') - (f(e) - ex_f(v))$
  $U'(e'') := U'(e'') - f(e)$

In this way an iteration operates on all loan account → security edges, first on all those with flows = capacity, then (iteratively) on all other edges where either

1. $f(e) = f'(e')$, i.e. flow on e = the not yet offset part of credit of a loan account or
2. $f(e) = U'(e'')$, i.e. flow on e = the not yet offset part of the security value
3. then again back (1., 2.) until the order of all assignments has been determined.

Network Preflow: Example

With a small network I will demonstrate here the way the preflow method operates.

The following figures show

- the capacities of the edges by the black numbers at the edges,
- the flows at the edges (total flow = flows minus reflows) are marked beneath in blue color
- the "distance labels" are indicated with 'D: *n*' at the corresponding vertices and
- the excess flow is indicated beneath 'ex: *nn*' (red color)
- the arrows at the arcs indicate the current flows

First flows are pushed from the source to the loan account vertices (left side of bipartite graph) according to the capacity of the edges. Fig.11 shows these flows and the excess flows at the loan account vertices.

After these initial pushes all loan account vertices are active. As all edges to the security vertices are admissible the excess flows are pushed at the first admissible edge leaving each active loan account vertex - which happen to be the edges to security vertex S1 for all loan account vertices. All these edges have sufficient capacity and no more loan account vertex will be active after these pushes.

As you see in Fig. 12 the capacity of the edge from security vertex S1 to the sink is only 24. Therefore a flow of 24 is pushed to the sink and the excess flow at vertex S1 is (48 - 24 =) 24. Vertex S1 will be active now (for reflow).

The reflow discharge operation manages the excess flow at vertex S1. As no admissible edges for the reflow (from loan account vertices to vertex S1) exists, S1 has to be relabeled. The new label of S1 is (2 + 1 =) 3, because all loan account vertices with edges to S1 (edges with "flow that can be pushed back") have a label value of 2: 2 is the "minimal label value of adjacent vertices" with "reverse capacity" - i.e. "flow to push back".

As Fig. 13 shows a part of the excess flow is pushed back to vertex L1, the other part to vertex L2 - according to the "reverse capacities" of these edges. After these reflows from vertex S1 the loan account vertices L1 and L2 have excess flows and are active.

Flows have now to be pushed again from the active loan account vertices to security vertices. The first edge of both active vertices (L1 and L2) is the edge to vertex S1 - but this edge is not admissible because the label value of the active vertex (L1 and L2 respectively) should be [label value of the adjacent vertex] + 1. In both cases, L1 and L2, the edges to S2 are admissible and the residual capacity is sufficient for the push of all the excess flow of each vertex.

Fig. 14 shows that a flow of 15 can be pushed from vertex S2 to the sink, and that there is now an excess flow of 9 at security vertex S2.

S2 is active now. But the excess flow cannot be pushed back because there is no admissible edge for the reflow.

Therefore S2 is relabeled, the new label value of S2 is 3 (see Fig. 15).
After relabeling S2 there are two admissible edges now: the edge from L1 and the edge from L2. The edge from L1 would have been chosen now for the reflow if there were not the special constraint on edge selection. But because of this constraint (see 3.5.5.3.2.1) the excess flow is pushed along the other edge, back to loan account vertex L2 (see Fig. 15).

Loan account vertex L2 has excess flow of value 9 and is active now.

At this moment only the edge leading to S3 is admissible. But according to the (residual) capacity of this edge no more than a flow of 5 can be pushed to S3 (see Fig. 16). So the vertex L2 is still active after this push - with excess flow of value 4. L2 has now to be relabeled because there are no more admissible edges with residual capacity. The new label value of L2 will be 4, and after this relabeling the two edges leading to S1 and S2 would be admissible, but only the edge to S1 has free residual capacity. The remaining excess flow of vertex L2 is pushed to vertex S1 - as is shown in Fig 16.

The security vertex S1 is active now and has excess flow of value 4.

As there is no flow on the edge from L1 to S1, the only admissible edge (with sufficient flow value for the reflow from S1) will be the edge from vertex L3 - and so the excess flow of S1 will flow (back) to vertex L3.

In this way the flows will spread across the network until the flow is optimized.

Abstract

Description of a method and software product to optimize the order of securities assignments to collateralized commercial loans in a network where each loan is secured by distinct securities and each collateral security balances distinct loan accounts assigned to these securities, thus forming a network that can be represented by a bipartite graph.

The method consists of

- 1st a network flow to optimize the amount of total secured credit
- 2nd a method to derive an optimal order from the values of the optimized network flow. The result is the order in which each assignment of distinct securities to distinct loan accounts is considered, so that the amount of total unsecured credit is minimized if loan accounts are balanced in this order.

loan accounts	*coll. securities*
L_1 *(1.0)*	S_1
	S_2
L_2 *(2.0)*	S_1
	S_2
	S_3
L_3 *(1.0)*	S_1
	S_3
	S_4
L_4 *(0.3)*	S_1
	S_3
	S_4
L_5 *(0.5)*	S_1

coll. securities - values :
S_1 *(2.4)*
S_2 *(1.5)*
S_3 *(0.5)*
S_4 *(0.3)*

Fig. 1 (values: million US $)

(represented as a network in *Fig. 2*)

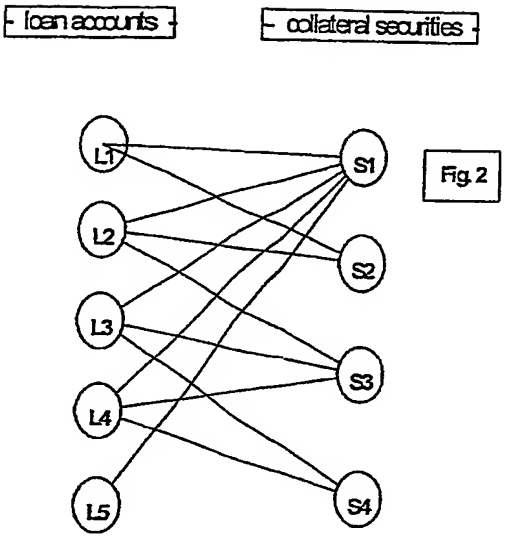

loan accounts
collateral securities
L1
L2
L3
L4
L5
S1
S2
S3
S4

Fig. 2

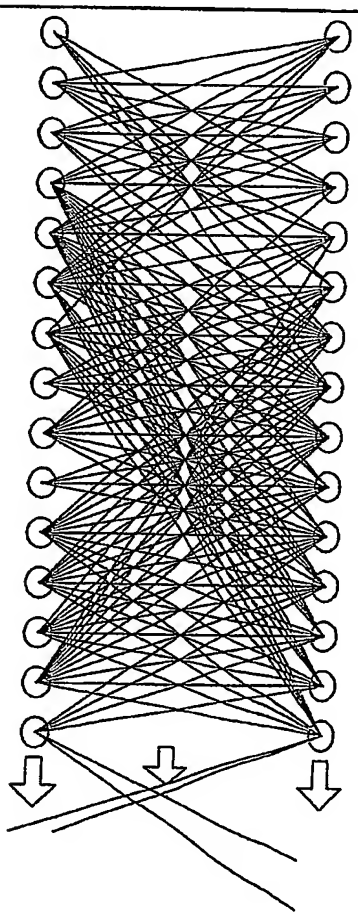


Fig. 3

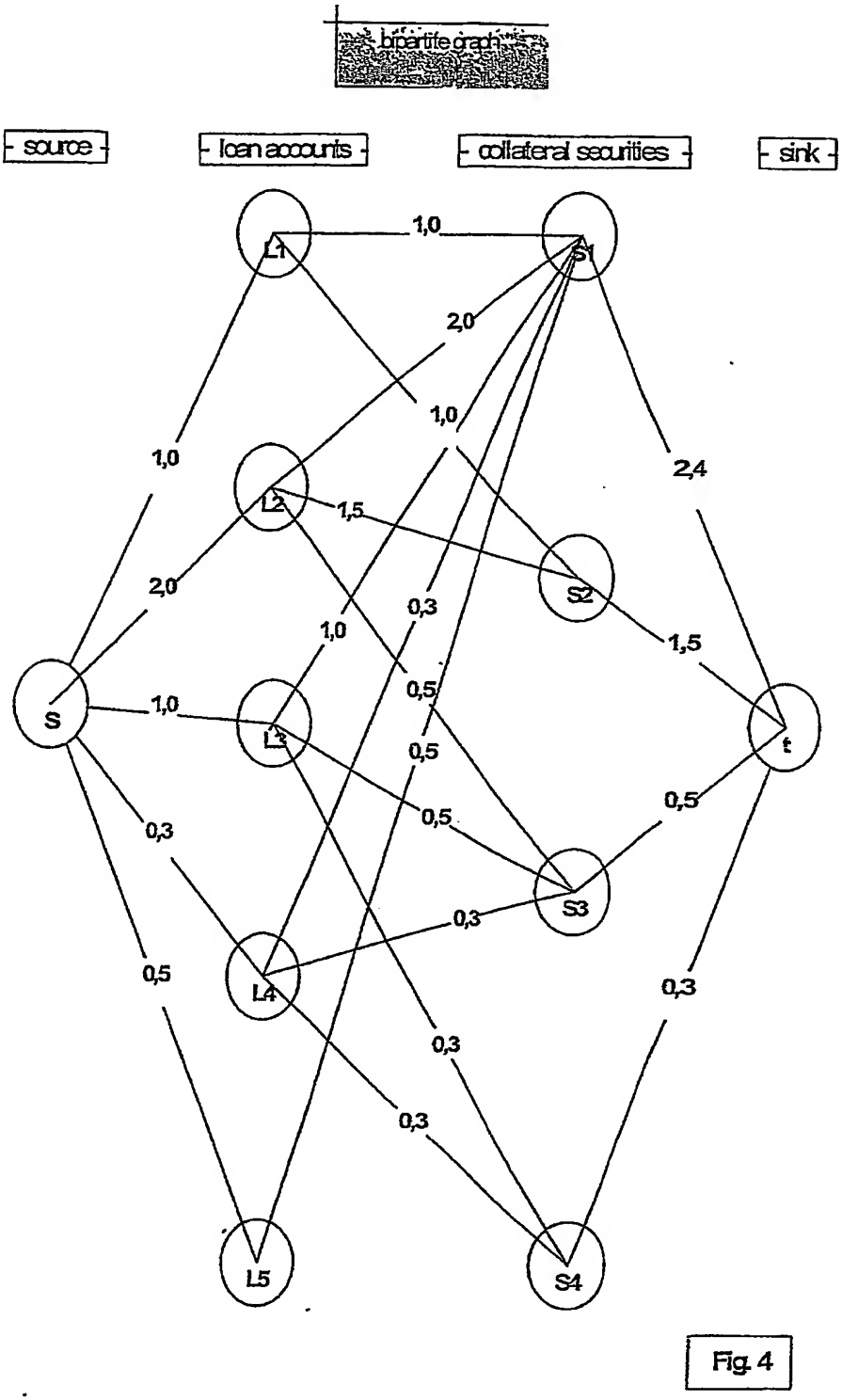

bipartite graph
source
loan accounts
collateral securities
sink
L1
L2
L3
L4
L5
S1
S2
S3
S4
S
t
1,0
2,0
1,0
1,0
2,4
1,5
2,0
0,3
1,0
1,5
0,5
1,0
0,5
0,5
0,5
0,3
0,3
0,5
0,3
0,3
0,3

Fig. 4

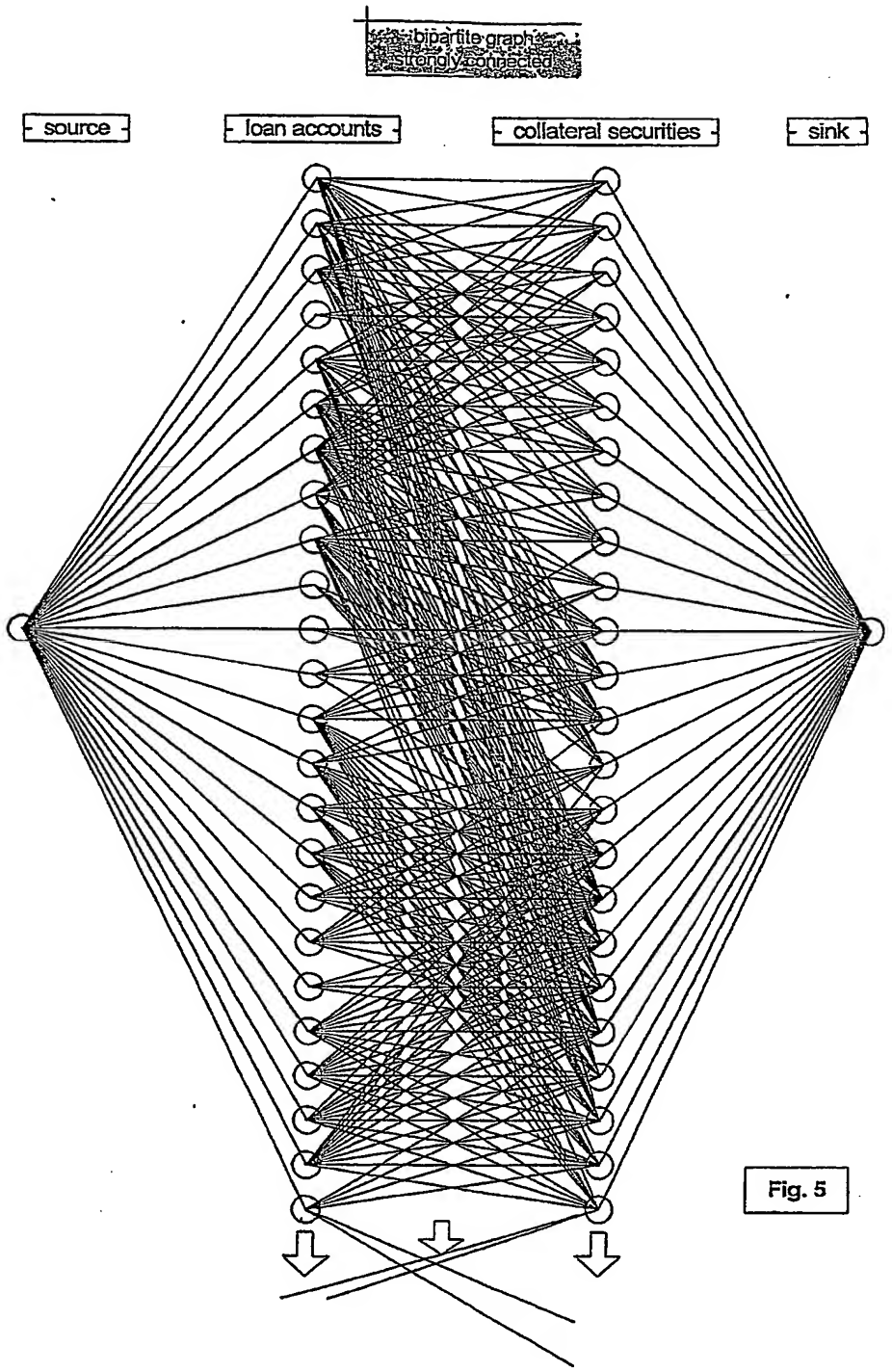

bipartite graph
strongly connected
source
loan accounts
collateral securities
sink

Fig. 5

s-t-cuts

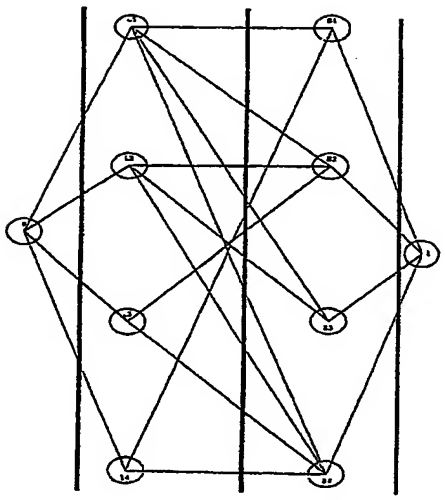


Fig. 6

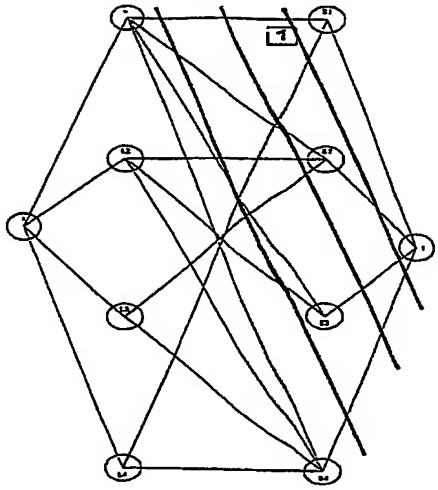


Fig. 7

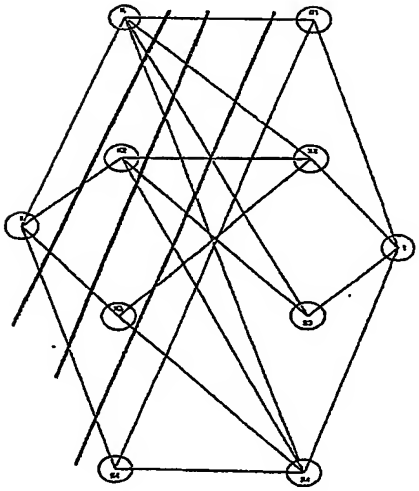


Fig. 8

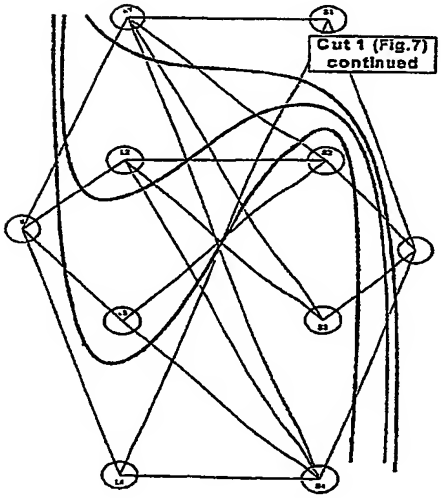
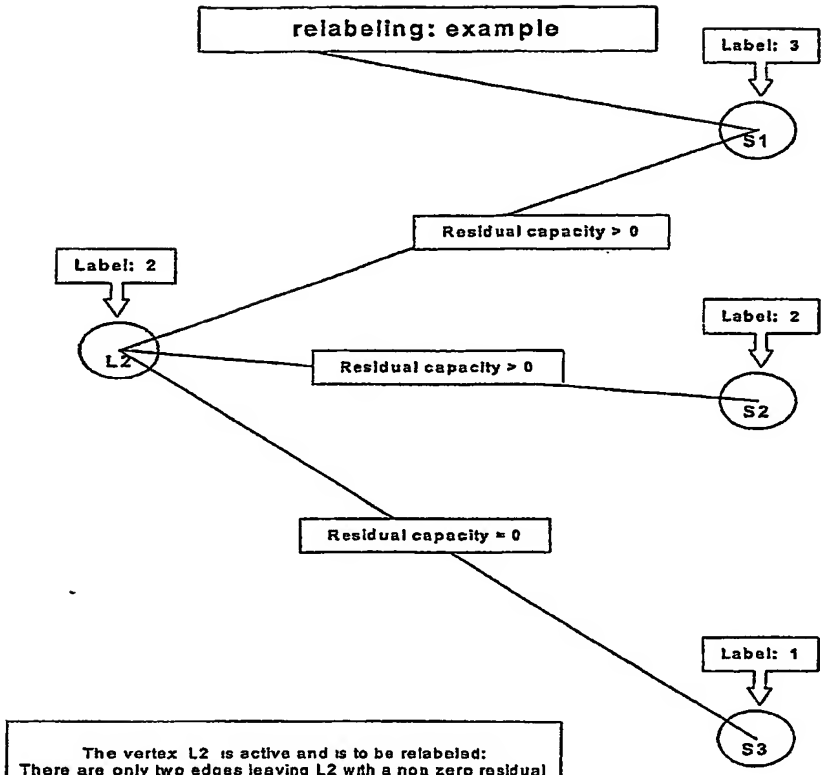


Fig. 9



The vertex L2 is active and is to be relabeled:
There are only two edges leaving L2 with a non zero residual capacity: the edges leading to the vertices S1 and S2. The smallest label of those two vertices will be the basis for the relabeling operation of vertex L2: The new label will be the value of [label of S2] + 1

new label of L2 = 2 + 1 = 3

Fig. 10

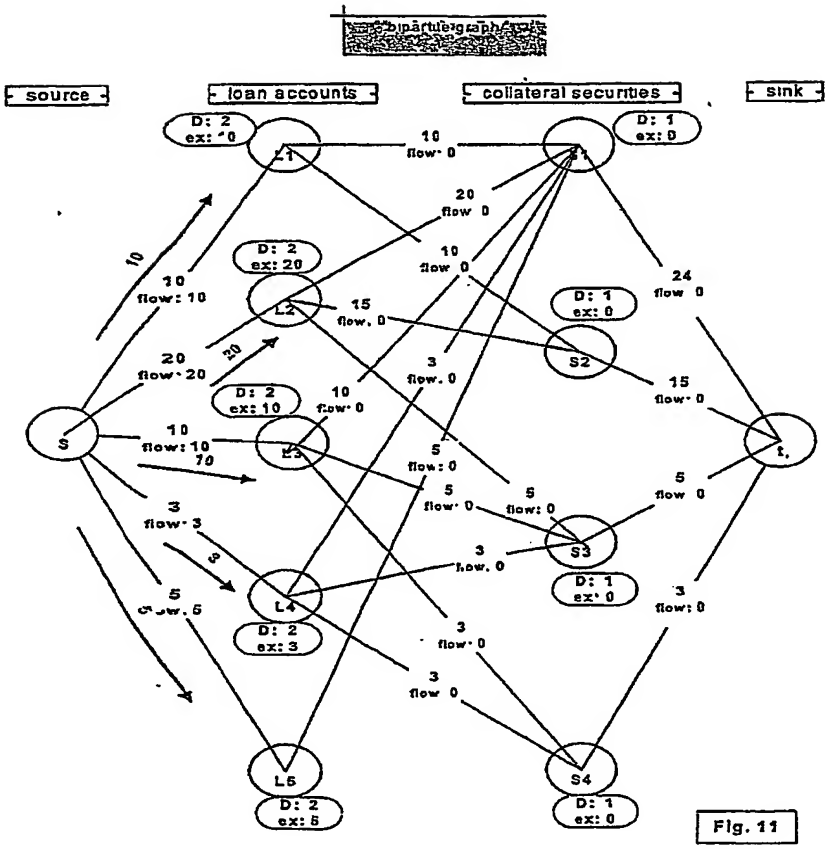

source
loan accounts
collateral securities
sink
D: 2
ex: 10
L1
10
flow: 0
D: 1
ex: 0
20
flow 0
D: 2
ex: 20
L2
10
flow 0
24
flow 0
10
flow: 10
15
flow. 0
D: 1
ex: 0
S2
20
flow: 20
20
3
flow. 0
15
flow: 0
D: 2
ex: 10
10
flow: 0
S
10
flow: 10
70
L3
5
flow: 0
t,
5
flow: 0
5
flow: 0
5
flow 0
3
flow: 3
S3
3
flow. 0
D: 1
ex: 0
L4
D: 2
ex: 3
3
flow: 0
3
flow: 0
3
flow 0
L5
D: 2
ex: 5
S4
D: 1
ex: 0

Fig. 11

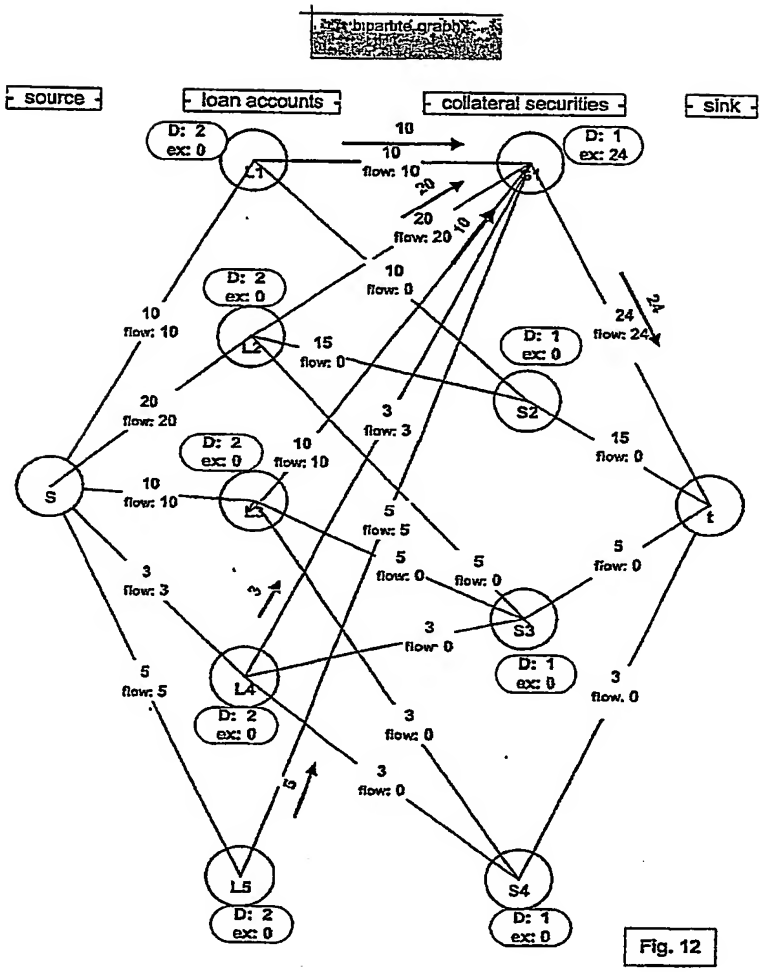

bipartite graph
source
loan accounts
collateral securities
sink
D: 2
ex: 0
L1
L2
L3
L4
L5
S
D: 1
ex: 24
S1
S2
S3
S4
t
10
flow: 10
20
flow: 20
flow: 0
15
flow: 0
3
flow: 3
5
flow: 5
24
flow: 24
D: 1
ex: 0

Fig. 12

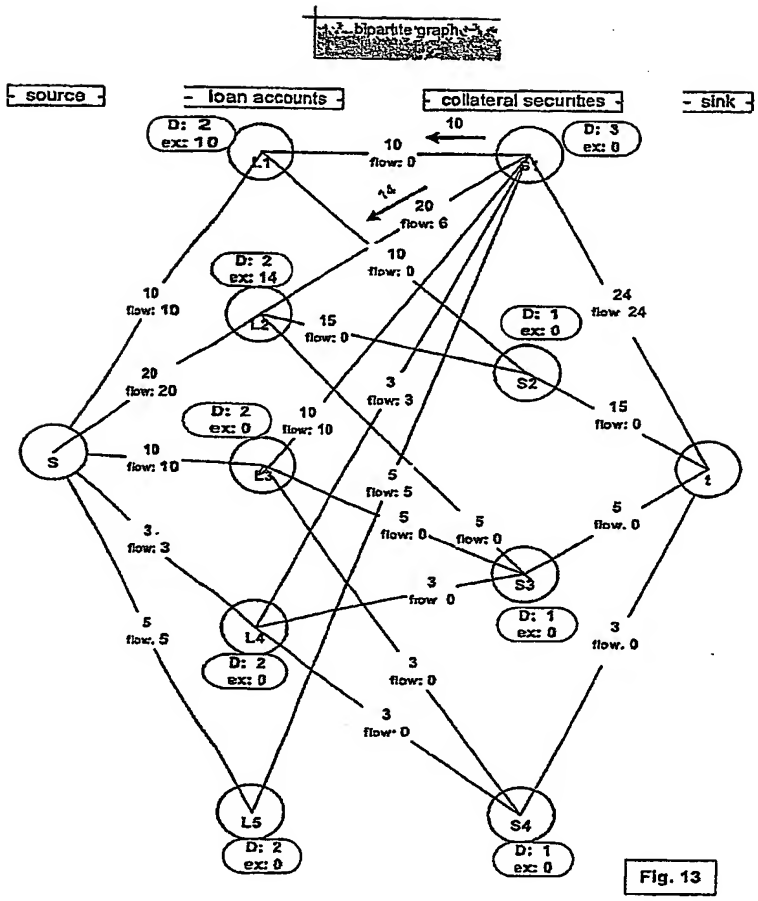

bipartite graph
source
loan accounts
collateral securities
sink
D: 2
ex: 10
L1
10
flow: 0
S1
10
D: 3
ex: 0
20
flow: 6
D: 2
ex: 14
10
flow: 0
L2
10
flow: 10
15
flow: 0
D: 1
ex: 0
24
flow 24
S2
20
flow: 20
3
flow: 3
D: 2
ex: 0
10
flow: 10
15
flow: 0
S
10
flow: 10
L3
5
flow: 5
t
5
flow: 0
5
flow: 0
5
flow. 0
3.
flow: 3
3
flow 0
S3
D: 1
ex: 0
5
flow. 5
L4
D: 2
ex: 0
3
flow. 0
3
flow: 0
3
flow: 0
L5
D: 2
ex: 0
S4
D: 1
ex: 0

Fig. 13

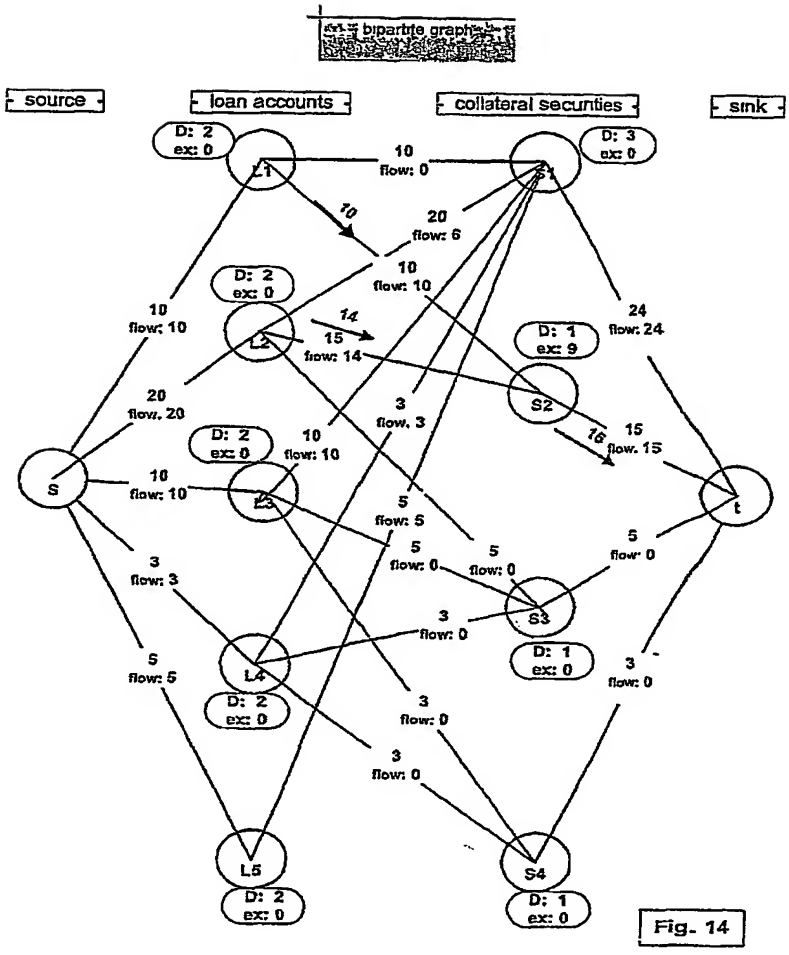

bipartite graph
source
loan accounts
collateral securities
sink
D: 2
ex: 0
L1
10
flow: 0
S1
D: 3
ex: 0
10
20
flow: 6
10
flow: 10
D: 2
ex: 0
L2
14
15
flow: 14
D: 1
ex: 9
24
flow: 24
10
flow: 10
20
flow: 20
S2
3
flow: 3
D: 2
ex: 0
10
flow: 10
15
flow: 15
15
S
10
flow: 10
L3
t
5
flow: 5
5
flow: 0
5
flow: 0
5
flow: 0
3
flow: 3
S3
3
flow: 0
D: 1
ex: 0
5
flow: 5
L4
3
flow: 0
D: 2
ex: 0
3
flow: 0
3
flow: 0
L5
S4
D: 2
ex: 0
D: 1
ex: 0

Fig. 14

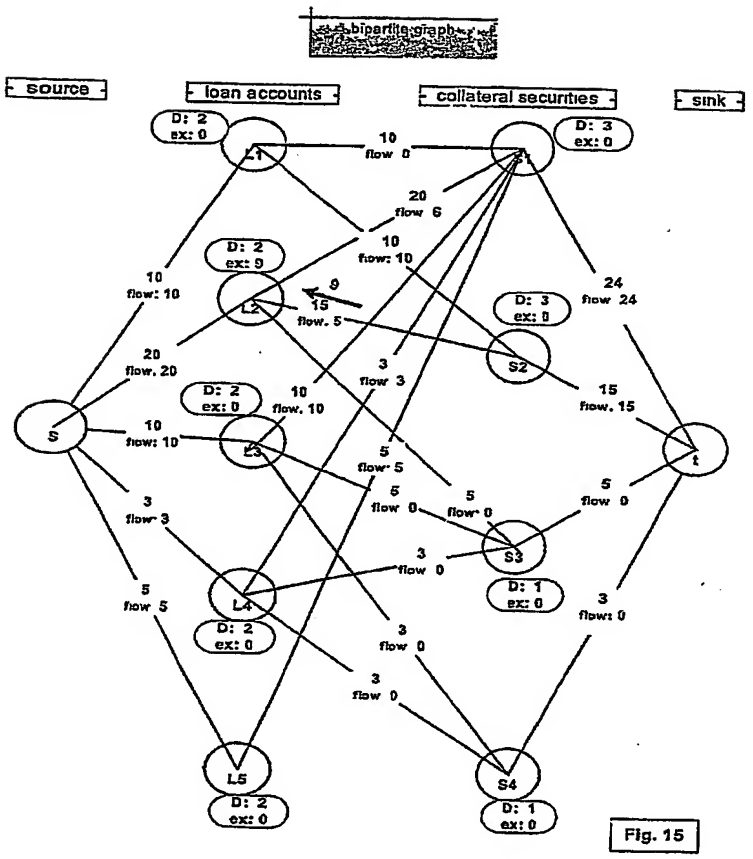

bipartite graph
source
loan accounts
collateral securities
sink
D: 2
ex: 0
L1
10
flow 0
S1
D: 3
ex: 0
20
flow 6
D: 2
ex: 9
10
flow: 10
10
flow: 10
L2
9
15
flow. 5
D: 3
ex: 0
24
flow 24
S2
20
flow. 20
3
flow 3
D: 2
ex: 0
10
flow. 10
15
flow. 15
s
10
flow: 10
L3
5
flow 5
t
5
flow 0
5
flow 0
5
flow 0
3
flow 3
S3
3
flow 0
D: 1
ex: 0
L4
D: 2
ex: 0
5
flow 5
3
flow 0
3
flow 0
3
flow 0
L5
D: 2
ex: 0
S4
D: 1
ex: 0

Fig. 15

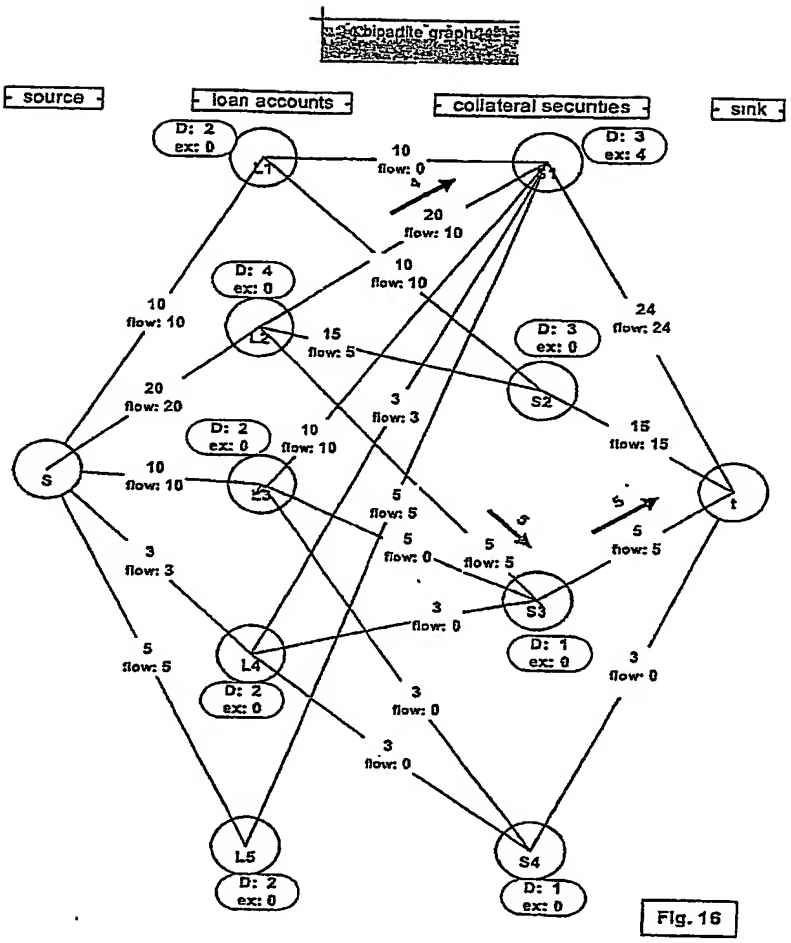

bipartite graph24
source
loan accounts
collateral securities
sink
D: 2
ex: 0
L1
10
flow: 0
S1
D: 3
ex: 4
20
flow: 10
D: 4
ex: 0
10
flow: 10
10
flow: 10
L2
15
flow: 5
D: 3
ex: 0
24
flow: 24
20
flow: 20
S2
3
flow: 3
D: 2
ex: 0
10
flow: 10
15
flow: 15
S
10
flow: 10
L3
5
flow: 5
t
5
flow: 0
5
flow: 5
5
flow: 5
3
flow: 3
3
flow: 0
S3
5
flow: 5
L4
D: 2
ex: 0
D: 1
ex: 0
3
flow: 0
3
flow: 0
3
flow: 0
L5
S4
D: 2
ex: 0
D: 1
ex: 0

Fig. 16